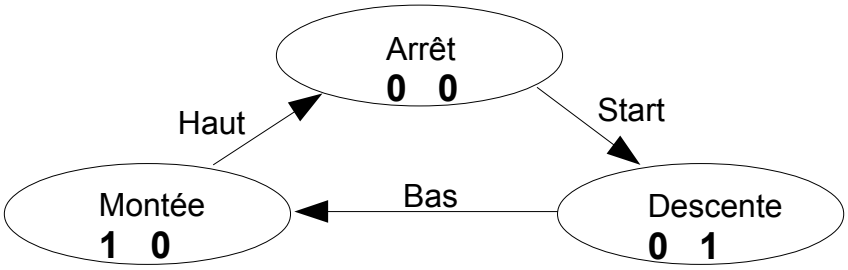
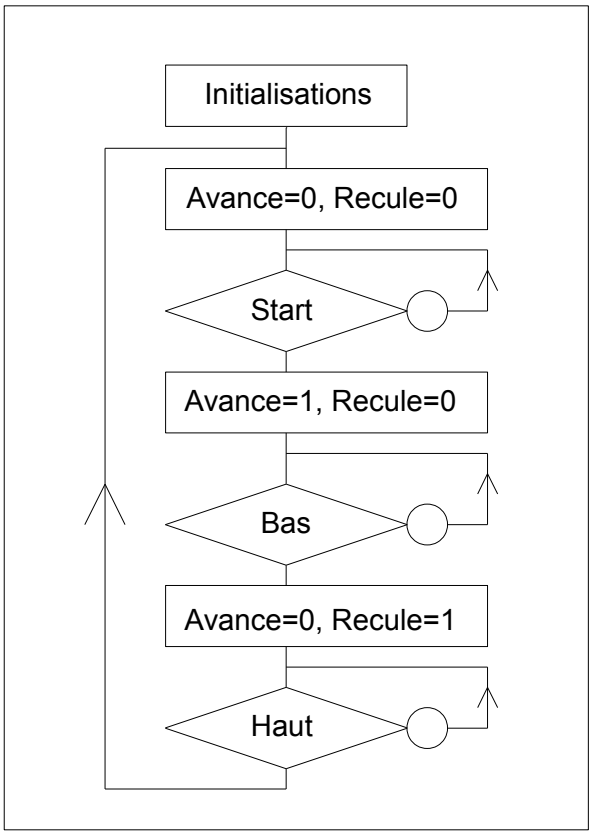
Programmation d'une machine d'état en C

Les structures de contrôle d'un langage de programmation, telles que les *if* et les *while*, permettent d'écrire un programme dont le fonctionnement est décrit par un graphe d'état. Reprenons l'exemple simple de la perceuse semi-automatique, dont voici le graphe d'état :



L'organigramme suivant réalise la machine d'état.
Le programme correspondant en C est très simple:




```
#define Start (!(digitalRead(P1_3)))
#define Bas (!(digitalRead(P2_7)))
#define Haut (!(digitalRead(P2_6)))
#define AvanceOn digitalWrite(P1_0,1)
#define AvanceOff digitalWrite(P1_0,0)
#define ReculeOn digitalWrite(P1_6,1)
#define ReculeOff digitalWrite(P1_6,0)

void setup() {
  pinMode(P1_0, OUTPUT);
  pinMode(P1_6, OUTPUT);
  pinMode(P1_3, INPUT_PULLUP);
  pinMode(P2_6, INPUT_PULLUP);
  pinMode(P2_7, INPUT_PULLUP);
}
void loop() {
  AvanceOff; ReculeOff;
  while(!Start) {
  }
  AvanceOn;
  while(!Bas) {
  }
  AvanceOff; ReculeOn;
  while(!Haut) {
  }
}
```

Avec cette manière de procéder, l'état du système est en fait mémorisé dans le compteur ordinal (*PC=programm counter*) du processeur. Par exemple, l'état descente correspond à l'exécution des instructions liées aux troisième *while* du programme. Il faut bien noter que dans ce programme, le boucle infinie `while(1)` correspond au parcours complet du graphe d'état.

Lorsqu'un graphe d'état devient complexe, il devient très vite compliqué de trouver l'organigramme et d'écrire le programme correspondant. De plus, une petite modification du graphe d'état peut avoir une grande incidence sur le programme. On préférera alors utiliser une technique très simple, utilisant une variable pour mémoriser l'état du système. Dans notre exemple, voici le programme correspondant. Seule la partie principale du programme est présentée :

```
char etat = 0;
void loop() {
      switch (etat) {
      case 0:     AvanceOff; ReculeOff;
                  if (Start) etat = 1; break;
      case 1:     AvanceOn; ReculeOff;
                  if (Bas) etat = 2; break;
      case 2:     AvanceOff; ReculeOn;
                  if (Haut) etat = 0; break;
      }
}
```

On notera que, contrairement au programme précédent, aucune boucle `while` ne se trouve à l'intérieur de la boucle infinie (`loop(1)`. Cela signifie que cette boucle s'exécute un très grand nombre de fois par seconde (quelques centaine de milliers avec un processeur MSP430 à 16 MHz).

Marche à suivre : La programmation d'une machine d'état, à partir d'un graphe d'état même très compliqué, peut s'écrire de la manière suivante:
- après les initialisations , une boucle infinie contient une instruction `switch` avec un `case` correspondant à chaque état.
- dans chaque `case`, on programme les valeurs des sorties du système associés à cet état.
- chaque transition partant d'un état est ensuite réalisée en plaçant dans le `case` correspondant une instruction `if` avec la condition logique sur les entrées associées à cette transition, suivie de l'assignation de la variable `etat` avec sa nouvelle valeur.

L'utilisation de valeurs numérique pour les état ne rend pas le programme lisible. L'utilisation de noms clairs est facile :

```
enum {Arret, Descente, Montee};
```